Name of Student: Pushkar Sane	
Roll Number: 45	Lab Assignment Number: 7
Title of Lab Assignment: To implement GUI programming and Database Connectivity.	
DOP:	DOS:

CO Mapped: CO4	PO Mapped: PO3, PO5, PSO1, PSO2	Signature:	Marks:

Practical No. 7

Aim: To implement GUI programming and Database.
1. To Design Login Page.
2. To Design Student Information Form/Library management Form OR Hospital Management Form.
3. Implement Database connectivity For Login Page i.e. Connect Login GUI with Sqlite3.

Description:

Graphical User Interface (GUI) programming in Python allows developers to create user-friendly applications with visual components. Python provides several GUI libraries, with Tkinter being one of the most popular and built into the standard library. Here are some key points on GUI programming with Python:

1. **Choice of Libraries:** Python offers various GUI libraries, including Tkinter, PyQt, PyGTK, and Kivy. Tkinter is a common choice for beginners and is widely used for building desktop applications due to its simplicity.

2. **Creating a GUI Application:** To create a GUI application, you need to create a main application window and add visual elements like buttons, labels, text entry fields, and more.

3. **Event Handling:** GUI programming involves event-driven programming. You define functions (callbacks) that are executed when specific events, like button clicks or mouse movements, occur.

4. **Layout Management:** GUI components need to be organized in a layout. Common layout managers include grid layout, pack layout, and place layout. They help you position components within the window.

5. **User Interaction:** GUIs provide a way for users to interact with the program. This includes input via buttons, text entry, and file dialogs, as well as feedback through labels and message boxes.

6. **Cross-Platform Development:** Python's GUI libraries are often cross-platform, allowing you to develop applications that work on Windows, macOS, and Linux.

7. **Graphics and Styling:** GUI programming allows you to customize the appearance of your application, including fonts, colors, and graphics.

8. **Database Connectivity:** You can connect your GUI application to databases like SQLite, MySQL, or PostgreSQL to store and retrieve data.

9. **Testing and Debugging:** GUI applications require thorough testing, especially for user interactions. Debugging GUIs often involves understanding event flow and component behavior.

Components Used in Designing Login Page and Student Information Form:

1. **Labels:** Labels are used to display static text or descriptions in the GUI. For example, "Username" and "Password" labels in a login page.

2. **Entry Widgets:** Entry widgets allow users to input data. In a login page, these are typically used for username and password input.

3. **Buttons:** Buttons are interactive elements that trigger actions when clicked. A "Login" button is used to initiate the login process.

4. **Message Boxes:** Message boxes are used to display information, alerts, or error messages. They are often used to confirm successful login or display login failures.

5. **Layout Managers:** Grid layout, pack layout, and place layout are used to organize and position the GUI components within the window.

Database Connectivity with SQLite3:

1. **SQLite Database:** SQLite is a lightweight, serverless, and self-contained database engine. It's often used for local data storage in desktop applications.

2. **Database Connection:** To connect to an SQLite database, you create a connection object using `sqlite3.connect('database_name.db')`.

3. **Cursor:** A cursor is used to execute SQL queries and fetch data from the database. You can use the cursor to create tables, insert data, and retrieve data.

4. **SQL Queries:** SQL queries are used to interact with the database. You can perform operations like creating tables, inserting records, selecting data, and updating or deleting records.

5. **Commit and Close:** After executing SQL statements, it's essential to commit changes to the database using `conn.commit()` and then close the connection with `conn.close()` to release resources.

Connecting the GUI with SQLite3 involves using SQL queries to check user credentials, typically during a login operation, as demonstrated in the previous responses. This allows you to verify the user's identity against a database and manage user data.

1. **To Design Login Page.**

   Code:

```python
import tkinter as tk
from tkinter import messagebox
# Function to check login credentials
def login():
    username = entry_username.get()
    password = entry_password.get()

    # Replace this with your own authentication logic
    if username == "Test" and password == "Test123":
        messagebox.showinfo("Login Status", "Login Successful")
    else:
        messagebox.showerror("Login Status", "Login Failed")

# Create the main window
window = tk.Tk()
window.title("Login Page")

# Create labels for username and password
label_username = tk.Label(window, text="Username:")
label_password = tk.Label(window, text="Password:")

# Create entry widgets for user input
entry_username = tk.Entry(window)
entry_password = tk.Entry(window, show="*") # Show '*' for password input

# Create a login button
login_button = tk.Button(window, text="Login", command=login)

# Place widgets on the window using the grid layout
label_username.grid(row=0, column=0)
entry_username.grid(row=0, column=1)
label_password.grid(row=1, column=0)
```

```
entry_password.grid(row=1, column=1)
login_button.grid(row=2, columnspan=2)


# Start the Tkinter main loop
window.mainloop()
```

Conclusion:

Designing a login page using a GUI in Python is a critical step in creating user-friendly applications. Key points include selecting a GUI library like Tkinter, using visual components (labels, entry fields, buttons), event-driven programming, layout management, user interaction elements, cross-platform compatibility, customization, database connectivity, and the importance of testing and security. A well-designed login page provides a secure and user-friendly entry point for applications.

Output:

Login Successful
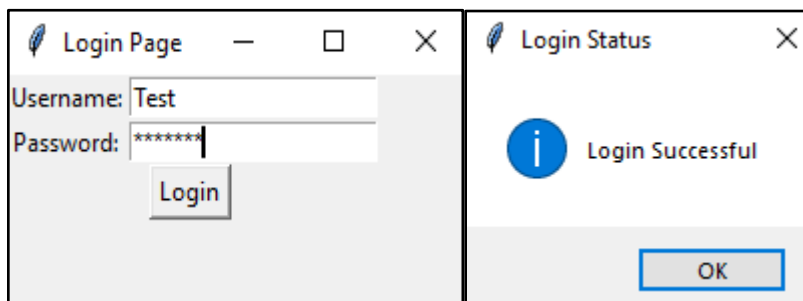
Login Failed
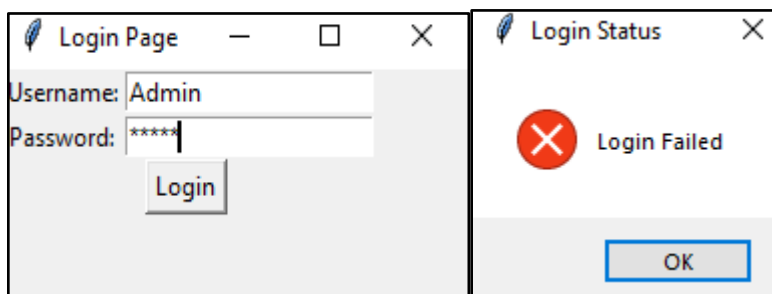
2. To Design Student Information Form.

Code:

```python
import tkinter as tk
def save_student_info():
    # Get data from input fields and save to a database or file
    name = entry_name.get()
    roll_number = entry_roll.get()

    # Add data processing logic here
    print(f"Saved Student: {name}, Roll: {roll_number}")

# Create the main window
window = tk.Tk()
window.title("Student Information Form")

# Create labels and entry widgets for student data
label_name = tk.Label(window, text="Name:")
label_roll = tk.Label(window, text="Roll Number:")
entry_name = tk.Entry(window)
entry_roll = tk.Entry(window)
save_button = tk.Button(window, text="Save", command=save_student_info)

# Place widgets on the window using the grid layout
label_name.grid(row=0, column=0)
entry_name.grid(row=0, column=1)
label_roll.grid(row=1, column=0)
entry_roll.grid(row=1, column=1)
save_button.grid(row=2, columnspan=2)
window.mainloop()
```
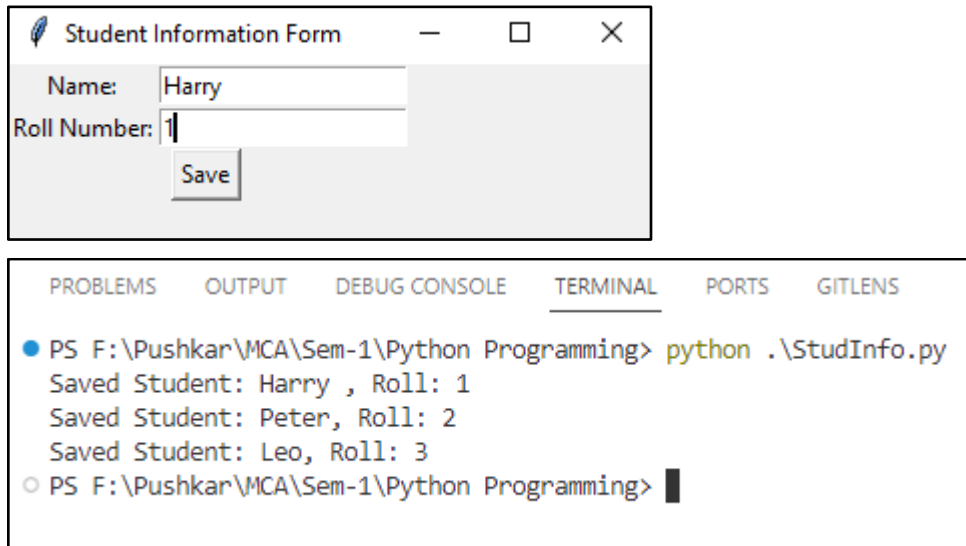
Conclusion:

Designing a Student Information Form using a GUI in Python allows for efficient data collection and management. It simplifies user input, streamlines data processing, and enhances user experience, making it an essential component for educational institutions and data-driven applications.

Output:

3. **Implement Database connectivity For Login Page i.e. Connect Login GUI with Sqlite3.**

Code:

```
import tkinter as tk
import sqlite3
from tkinter import messagebox

# Function to check login credentials
def login():
    username = entry_username.get()
    password = entry_password.get()

    # Connect to the SQLite database
    conn = sqlite3.connect("user_credentials.db")
```

```
    cursor = conn.cursor()


    cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", ("Test",
"Test123"))
    cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)",
("Pushkar", "Admin"))
    cursor.execute('''CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY,
        username TEXT NOT NULL,
        password TEXT NOT NULL
    )''')


    # Check if the user exists in the database
    cursor.execute("SELECT * FROM users WHERE username=? AND password=?",
(username, password))
    user = cursor.fetchone()
    if user:
        messagebox.showinfo("Login Status", "Login Successful")
    else:
        messagebox.showerror("Login Status", "Login Failed")


    # Close the database connection
    conn.close()

# Create the main window
window = tk.Tk()
window.title("Login Page")

# Create labels for username and password
label_username = tk.Label(window, text="Username:")
label_password = tk.Label(window, text="Password:")

# Create entry widgets for user input
entry_username = tk.Entry(window)
```

entry_password = tk.Entry(window, show="*") # Show '*' for password input

# Create a login button
login_button = tk.Button(window, text="Login", command=login)

# Place widgets on the window using the grid layout
label_username.grid(row=0, column=0)
entry_username.grid(row=0, column=1)
label_password.grid(row=1, column=0)
entry_password.grid(row=1, column=1)
login_button.grid(row=2, columnspan=2)

# Start the Tkinter main loop
window.mainloop()

Conclusion:

Connecting a Login GUI with SQLite3 database enables secure user authentication. It enhances data storage and retrieval capabilities, ensuring user credentials are validated effectively, making it a crucial feature for login systems in Python applications.

Output:

User Available In sqlite3 Database
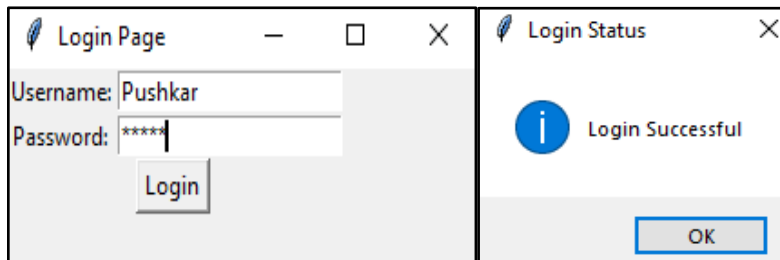
User Not Available In sqlite3 Database